Lecture Notes for Sep 18, 2025

Homework

There's homework this weekend for some code-reading practice.

Professor's Blabbering

- the more code you read, the better your ability to read code gets (*no derrr*)
  - understanding the codebase is important, can only be done by reading code

You can lead a horse to water, but you can't make it drink.

proceeds to ask that we do the homework… like we haven't been already…

The Filter Procedure

The `filter` procedure takes a list and returns a **filtered** list.

yes, I made that a recursive definition. deal with it. you already know what "filter" means.

Here's how to use it:

```
(filter <t-f-procedure> <list>)

> (define my-list (list 1 2 3 4 5 6))
> (define (even? num) (= (remainder num 2)))
> (filter even? my-list)
'(2 4 6)
```

Data Types in Scheme

- **Numbers**: `1`, `10.8`, `4.5e3`
- **Characters/Strings**: `banana`, `b`
- **Booleans**: `#t`, `#f`
- **Vectors**: `#(1 2 3 "hi" 3.7)`
- **Procedures**
- **Pairs & Lists**: `(42 . 8)`, `(10 5 99 32 108)`
- **Symbols**: `pi`, `+`, `x`, `foo`, `hello-world`

Symbols

Data types that let us *pass* and *return* **symbols** are **First Class**.

You can get a symbol as a string *rather than evaluating it* with the **quote** procedure:

```
> (define x 5)
> x
5
> (quote x)
x
```